Deep Learning
Verfahren zur Lösung von Problemen im Bereich Computational Engineering

MANFRED KRAFCZYK
Institut für rechnergestützte Modellierung im Bauingenieurwesen, TU Braunschweig

Zum Status Quo

Computational Engineering hat sich in den letzten Jahrzehnten als ein interdisziplinäres Paradigma entwickelt, welches sich mit der Entwicklung und Anwendung von Rechenmodellen und Simulationen, oft unter Verwendung von Hochleistungsrechnern, beschäftigt, um komplexe Mehrfeldprobleme zu lösen, die bei der technischen Analyse und Konstruktion von Ingenieurproblemen auftreten. Die rechnergestützte Simulation bietet dabei die Möglichkeit, in Bereiche vorzudringen, die entweder für traditionelle Experimente unzugänglich sind oder in denen die Durchführung traditioneller empirischer Untersuchungen zu aufwändig ist. Die Entwicklung eines konkreten Simulationsmodells oder Verfahrens beginnt dabei typischerweise mit der Systemidentifikation, bei der ein System als ein Konglomerat individueller Entitäten und ihrer Wechselwirkung gedacht wird. Viele Ingenieursysteme insbesondere in der Struktur- und Festkörpermechanik zeichnen sich durch die Beschreibung von Systemen als Continua über sog. Felder aus, die miteinander in Raum und Zeit interagieren. Die mathematische Beschreibung erfolgt dann in Form von partiellen Differenzialgleichungen, die es unter Annahme geeigneter Rand- und Anfangsbedingungen zu lösen gilt. Durch den zumeist nicht-linearen Charakter dieser Differenzialgleichungen ist es nur in Ausnahmefällen möglich, analytische Lösungen zu finden. Alternativ werden seit vielen Jahrzenten numerische Verfahren entwickelt, die für endlich viele Punkte oder Intervalle einer Raum-Zeit-Partitionierung Näherungslösungen abzuleiten erlauben. Der resultierende Berechnungsaufwand für eine solche Näherungslösung hängt dabei sowohl von der gewünschten Genauigkeit als auch der algorithmischen Komplexität des zugrundeliegenden numerischen Verfahrens ab. Grundsätzlich besteht ein nichtlinearer, aber monotoner Zusammenhang zwischen Berechnungsaufwand und erzielbarer Genauigkeit, d.h. je genauer die Näherungslösung sein soll, desto aufwändiger ist die Berechnung. Während bei der Lösung eines eindimensionalen Problems im Raum eine Verdopplung der gewünschten Genauigkeit im Wesentlichen maximal zu einer Verdopplung des Rechenaufwandes führt, erhöht die Anzahl der räumlichen Dimensionen die Anzahl der zu berechnenden Unbekannten exponentiell, d.h. in drei Raumdimensionen führt dies zur Erhöhung des Rechenaufwandes um eine Größenordnung. Da neben der Anzahl der Dimensionen in Raum und Zeit oft noch weitere Parameter im Sinne einer Problemdimension variiert werden, sind wir sehr oft mit den Auswirkungen des sogenannten Fluches der Dimensionen [1] konfrontiert, der für viele praxisrelevante Probleme zu inakzeptablen Rechenzeiten bei der Auffindung einer hinreichend genauen Näherungslösung führt. Neben dem Aspekt der Berechnungskomplexität spielt bei numerischen Näherungsverfahren die Frage der Stabilität und Konsistenz eine wesentliche Rolle. Stabilität bezieht sich dabei auf die Eigenschaft eines Verfahrens, dass unvermeidbare Störungen in Zwischenlösungen während der Berechnung nicht grenzenlos wachsen, sondern monoton gedämpft werden. Diese Stabilität kann für ein gegebenes Verfahren z.B. unter Verwendung einer Neumann-Analyse untersucht und optimiert werden. Konsistenz beschreibt den Umstand, dass im Falle asymptotisch kleiner Orts- und Zeitschrittweiten die numerische Lösung mit einer definierten Ordnung (polynomial oder exponentiell) gegen die unbekannte

https://doi.org/10.24355/dbbs.084-202109031135-0

wahre Lösung des Problems strebt. Das Lax-Wendrow Theorem beweist vereinfacht formuliert, dass Stabilität und Konsistenz hinreichende Bedingungen für die Konvergenz eines numerischen Verfahrens darstellen. An dieser Stelle ist festzuhalten, dass für hochentwickelte numerische Verfahren im Allgemeinen relativ klar ist, unter welchen Umständen sie mit welchem Berechnungsaufwand zu einer Näherungslösung mit konservativ abschätzbarer Genauigkeit in einer definierten Fehlernorm führen, insbesondere, wenn sie mit Methoden der *a posteriori* Fehlerschätzung gekoppelt werden. Grundsätzlich kann festgestellt werden, dass zwar prinzipiell viele leistungsfähige numerische Methoden (z.B. Finite Elemente, Finite Differenzen, Finite Volumen, Spektrale Methoden u.a.) zur Lösung von partiellen Differentialgleichungen im Kontext des Computational Engineering verfügbar sind und deren Eigenschaften bzgl. Robustheit und Genauigkeit abgesichert ist; die aus der Nutzung solcher Verfahren resultierenden Berechnungszeiten sind jedoch für viele praxisnahe Problemstellungen selbst bei Nutzung leistungsfähiger paralleler Hardware auf absehbare Zeit inakzeptabel hoch. Daher besteht weiterhin der Bedarf nach innovativen Methoden zur (näherungsweisen) Lösung von partiellen Differentialgleichungen.

Deep Learning

Obwohl erste begriffsprägende Publikationen über Maschinelles Lernen als Teilgebiet der sog. Künstlichen Intelligenz schon vor langer Zeit publiziert wurden (z.B. [12][13]), existiert eine Vielzahl von Definitionen, auf die man sich beziehen kann. In Anlehnung an [14] bezieht sich dieser Artikel auf Maschinelles Lernen als die Wissenschaft von rechnergestützten Methoden, die ihre Problemlösungskapazität durch sukzessive Verarbeitung größerer Datenmengen „selbständig“ erweitern. Unter dem Sammelbegriff des Maschinellen Lernens versammeln sich eine Vielzahl von methodischen Ansätzen, welche zu diskutieren den Rahmen dieses Berichtes übersteigt. Evident ist jedoch, dass die Methoden des Maschinellen Lernens in den letzten beiden Dekaden eine beispiellose Erfolgsgeschichte in Forschung und Technik geschrieben haben, da sie im Kontext der weltweit verfügbaren, exponentiell gewachsenen Datenflut in Verbindung mit leistungsfähiger Hardware mit neuartigen Problemlösungsstrategien unseren Umgang mit Daten und die Fähigkeit, aus Ihnen Informationen, Strukturen und Wissen abzuleiten, revolutioniert haben. Im Folgenden wird auf ein Teilgebiet des Maschinellen Lernens eingegangen, welches seit kurzem zur Berechnung von Näherungslösungen von partiellen Differentialgleichungen adaptiert wurde. Das sogenannte Deep Learning basiert auf der Nutzung von artifiziellen Neuronalen Netzen (ANN). Ein ANN ist schematisch in Abbildung 1 dargestellt und besteht aus diversen Schichten, die jeweils aus sog. Neuronen bestehen, welche Daten halten (in einfachster Form als Fließkommazahl). In der Eingangsschicht werden die Eingabedaten in einer geeigneten Form kodiert. Die Ausgabeschicht enthält die Berechnungsergebnisse aus der sukzessiven Transformation der Eingangsdaten über die sogenannten „versteckten“ Schichten, die jeweils wieder aus einer definierten Anzahl von Neuronen bestehen. Mathematisch betrachtet handelt es sich bei einem ANN um eine Abbildung zwischen zwei Räumen (beispielsweise reeller Zahlen) von potenziell unterschiedlicher Dimension. Gemäß dem *universal approximation theorem* [8][9] kann ein sog. vorwärtsbetriebenes neuronales Netz (engl. Feedforward Neural Network, FNN) mit einer linearen Ausgabeschicht und mindestens einer versteckten Schicht unter gewissen Annahmen bezüglich der zu verwenden Aktivierungsfunktion jede beliebige Funktion mit beliebig kleinem Fehlerbetrag approximieren.

https://doi.org/10.24355/dbbs.084-202109031135-0

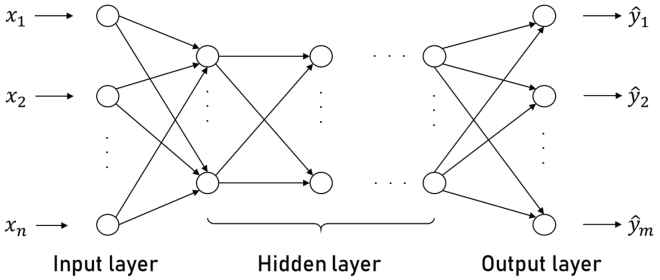


Abb. 1: Prinzipdarstellung eines „tiefen" neuronalen Netzes (nach [15])

Der Wert jedes einzelnen Neurons der Ausgangsschicht (i.e. Ergebnisschicht) hängt dabei potenziell von allen Zuständen der Neuronen aller vorangehenden Schichten inklusive der Eingangsschicht ab. Die Anzahl der zu wählenden verborgenen Schichten sowie deren Größe (i.e. Anzahl von Neuronen) in Verbindung mit einer nicht-linearen Aktivierungsfunktion sind weitgehend frei wählbar und bestimmen wesentlich die Flexibilität des ANN, komplexe nicht-lineare Abbildungen zwischen Eingangs- und Ausgangsschicht zu repräsentieren. Die nachfolgende Abbildung zeigt exemplarisch die Berechnung des Zustandes eines Neurons aus allen Neuronen der vorherigen Schicht. Dieses Schema setzt sich für alle vorherigen Schichten entsprechend fort. Die erste verborgene Schicht wird analog aus der Eingangsschicht berechnet. Zwischen je zwei Schichten fungieren nun sogenannte Wichtungsfaktoren, welche neben einem sog. Bias und einer sog. Aktivierungsfunktion als freie Parameter des ANN zu „lernen" sind. Bei k verborgenen Schichten mit je m Neuronen ergeben sich also maximal $1 + 3m + (k - 1)(m^2 + m)$ Freiheitsgrade, welche die Abbildung von der Eingangs- zur Ausgangsschicht beeinflussen.

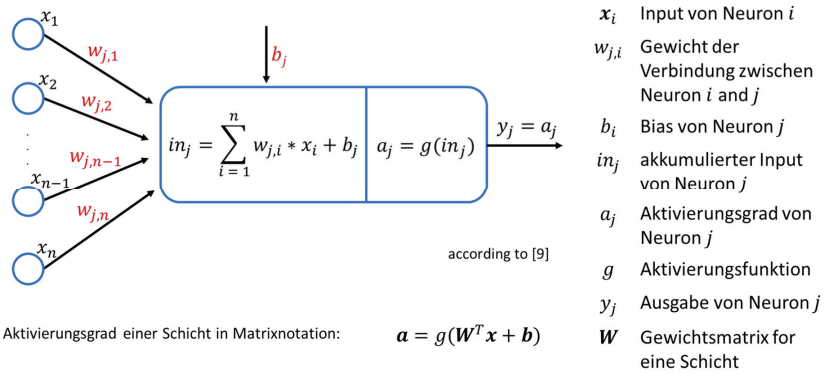


Abb. 2. Berechnung des Zustandes eines Ausgangsneurons aus den Eingangsneuronen (ohne verborgene Schichten) (nach [16])

Der wesentliche Punkt ist nun die Wahl der Freiheitsgrade (i.e. Wichtungsfaktoren, Bias-Werte, Aktivierungsfunktion(en)), so dass die Abbildung der Eingangsdaten auf die Ausgangsdaten eine Näherungslösung für eine spezifische Problemstellung entspricht. Als Beispielproblem sei

https://doi.org/10.24355/dbbs.084-202109031135-0

nachfolgend die Identifikation von Ziffern aus den Scans handschriftlicher Notierungen (i.e. optical character recognition, OCR) skizziert. In Abbildung 3 sehen wir unterschiedliche Beispiele von handschriftlichen Zahlendarstellungen, deren statistisch korrekte Zuordnung zu den zehn Basisziffern (0-9) einem Menschen mit rudimentärer Schulbildung relativ leichtfallen würde.

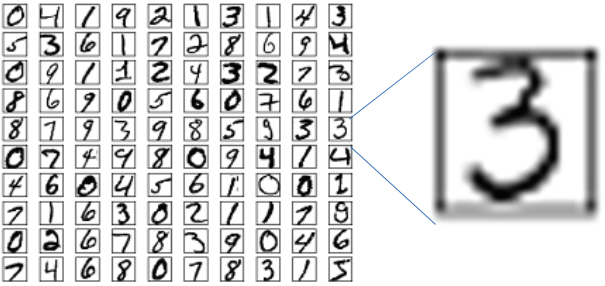


Abb. 3. Graustufencodierte scans handschriftlich notierter Ziffern[1]

Datentechnisch kann jedes einzelne Bild beispielsweise als eine Matrix aus Grauwerten repräsentiert werden, bei einer Bildgröße von 128x128 Pixeln lägen als Eingangswerte also 16.384 Grauwerte vor, die als Fließkommazahlen Werte zwischen 0 (schwarz) und 1 (weiß) annehmen. Unser ANN bildet also einen Eingangsvektor mit 16.384 Elementen im Extremfall auf einen einzelnen skalaren Wert (nämlich der identifizierten Ziffer) ab. Die Berechnung dieses Ausgangszustandes setzt nun voraus, dass eine konkrete Anzahl verborgener Schichten mit einer dedizierten Zahl von Neuronen und entsprechende Parametrisierungen für die Wichtungsfaktoren, Biaswerte etc. bestimmt werden. Die „beste“ Anzahl verborgener Schichten ist ebenso wie deren Größe *a priori* unklar. Für Probleme dieser Art können wenige Schichten mit einer höheren zweistelligen Anzahl von Neuronen schon sehr gute Ergebnisse liefern. Theoretisch können die Wichtungsfaktoren und Biaswerte beliebig initiiert werden.

Nachdem alle Parameter feststehen, wird die Abbildung des Eingangszustandes auf den Ausgangszustand jedoch keine statistisch signifikanten Erfolge bei der Identifikation der „richtigen“ Ziffer zeitigen. Für eine gegebene Menge an sog. Trainingsdatensätzen (für die das richtige Ergebnis der Klassifikation vorgegeben wird), kann so eine mittlere Abweichung der Vorhersage über alle Trainingsdaten quantifiziert werden. Die Abweichung wird auch als Loss-Funktion bezeichnet. Die Herausforderung besteht nun darin, die Freiheitsgrade des ANN (in der Regel Wichtungsfaktoren und Biaswerte) iterativ so zu bestimmen, dass die Loss-Funktion minimiert wird. Dazu wird über das sog. Automatische Differenzieren [6] quantifiziert, in welche Richtung sich die Freiheitsgrade individuell quantitativ ändern müssten, um die Loss-Funktion in einem gewissen Fehlermaß statistisch zu reduzieren. Diese Information fließt über die sog. Backpropagation von der letzten zur ersten verborgenen Schicht und modifiziert sukzessive alle Freiheitsgrade. Diese Optimierung wird über eine Vielzahl von Trainingsdatensätzen durchgeführt. Ein kompletter Durchlauf aller Input-Daten wird dabei jeweils als Epoche bezeichnet. Mit zunehmender Zahl an Epochen wird die Vorhersagefähigkeit für neue Testfälle statistisch zunehmen, um dann über ein Maximum ggf.

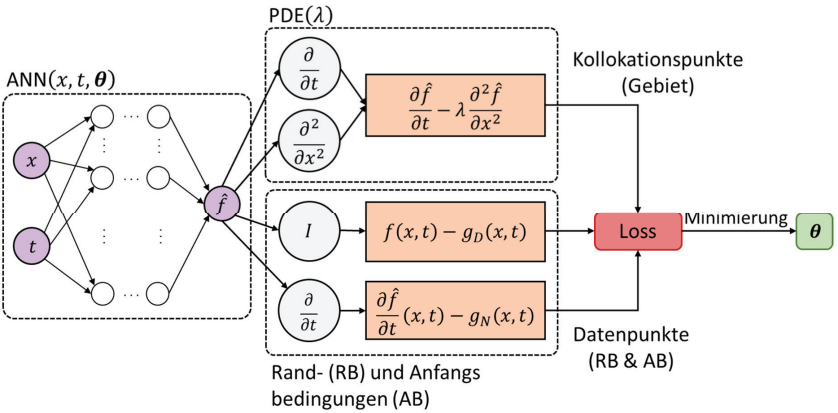
[1] http://neuralnetworksanddeeplearning.com/chap1.html

https://doi.org/10.24355/dbbs.084-202109031135-0

wieder tendenziell abzufallen. Die optimale Menge an Trainingsdaten sowie an Epochen können bisher nicht *a priori* vorhergesagt werden. Nach geeigneter Trainingsphase ist das ANN dann in der Lage, neue Datensätze mit relativ hoher Wahrscheinlichkeit korrekt zu kategorisieren. Es ist jedoch anzumerken, dass ein DNN bei der Identifikation von Mustern auch vorsätzlich „in die Irre“ geführt werden kann [11], indem das Eingangssignal auf subtile Weise gestört wird, wobei diese Störung die menschliche Charakterisierung nicht notwendigerweise beeinflussen würde. Solche Artefakte sind natürlich bei fortgeschrittenen Problemstellungen der Musteridentifikation wie dem sensorbasierten autonomen Fahren von viel größerer Bedeutung als beim OCR-Prozess, da sie zu situativen Fehleinschätzungen mit großen Folgeschäden führen können.

Physics Informed Neural Networks (PINNS)

In [3] wurde kürzlich basierend auf [4][5] ein ANN vorgestellt, das sich prinzipiell zur Berechnung von Näherungslösungen für partielle Differentialgleichungen eignet. Obwohl die Differentialgleichungen nicht notwendigerweise physikalische Systeme beschreiben müssen, hat sich in der Zwischenzeit der Begriff der *Physics informed Neural Networks* (PiNNS) für diese Modelle etabliert. Ruft man sich in Erinnerung, dass ein ANN im Wesentlichen konstruiert wird, um für ein bestimmtes Problem die resultierende Loss-Funktion zu minimieren, liegt es nahe, die Loss-Funktion über ein Residuum zu definieren, welches aus dem Einsetzen der iterativ berechneten Näherungsfunktion für einen Satz von Raum- und Zeitpunkten in die Differentialgleichung resultiert. Die nachfolgende Abbildung veranschaulicht das Konzept für eine Differenzialgleichung der Form $\frac{\partial f}{\partial t} = \lambda \frac{\partial^2 f}{\partial x^2}$, welche beispielsweise eine zeitabhängige Diffusion einer Konzentration f in einer Raumdimension beschreibt.



Die in [3][4][5] vorgestellten Beispiellösungen legen nahe, dass die Lösung partieller Differenzialgleichungen mit PINNs sehr effizient sein kann. Dies gilt insbesondere, da die PINNs so konzipiert werden können, dass die Abbildung durch das FNN komplexe numerische Methoden wie z.B. Runke-Kutta-Verfahren (RK) sehr hoher Ordnung (500) zur Lösung zeitabhängiger Probleme „imitieren“ kann, nachdem sie entsprechend trainiert wurden. Die durch das FNN oder PINN imitierte Berechnungsmethode benötigt allerdings (unter

https://doi.org/10.24355/dbbs.084-202109031135-0

Vernachlässigung der Trainingsphase) nur einen winzigen Bruchteil der Berechnungszeit wie das ursprüngliche klassische RK-Verfahren und erzielt ausgezeichnete Genauigkeiten.

Ausblick und Zusammenfassung

Die ersten Ergebnisse bei der Nutzung von tiefen neuronalen Netzen zur Lösung komplexer Probleme im Bereich des Computational Engineering stellen eine neue Klasse von Lösungsverfahren mit sehr hoher Effizienz in Aussicht. Trotzdem liegen noch keine hinreichenden Informationen zu einer abschließenden Bewertung vor. Obwohl fundamental klar ist, dass durch das *universal approximation theorem* [8][9] die grundsätzliche Mächtigkeit der Abbildungskomplexität vorhanden ist, sind noch grundsätzliche Fragen zur verlässlichen Nutzung von PINNs für komplexe Ingenieurprobleme zu klären. Dies umfasst zum einen die Frage, wie quantitativ die zu erzielende Genauigkeit einer PINN-basierten Näherungslösung von der Anzahl und Größe der verborgenen Schichten (und damit auch des assoziierten Trainings- und Berechnungsaufwandes) abhängt. Hier könnten auch klassische Methodenvarianten wie raumzeitliche Adaption basierend auf *a priori* Fehlerschätzern von sample-Punkten auf PiNNs übertragen werden und zu einer signifikanten Effizienzsteigerung beitragen. Während bei Überschreitung von Stabilitätsgrenzen klassische numerische Verfahren einfach versagen, tritt ein solcher Effekt bei PiNNs nicht ein. Dies kann bei unreflektiertem Einsatz von PiNNs dazu führen, dass die erzielten Lösungen von unzureichender Qualität sind. Weiterhin können wir klassischen numerischen Verfahren wie Finiten Elementen fast ausnahmslos eine definierte Konvergenzordnung zuschreiben, die den Berechnungsaufwand in Form von Anzahl der verwendeten Freiheitsgrade mit der zu erwartenden Ergebnisqualität zu korrelieren erlaubt. Inwieweit eine solche Korrelation bei PINNs auch ein robustes Methodenattribut darstellt und von welchen Parametern sie abhängt, ist nach Kenntnis des Autors zur Zeit dieser Niederschrift noch nicht grundlegend klar. Hier ist auch anzumerken, dass die Minimierung der Loss-Funktion niemals die Erreichung des globalen Minimums garantiert, sondern nur lokale Minima gefunden werden (von denen es im Suchraum beliebig viele geben kann).

Neben der Lösung von partiellen Differenzialgleichungen im Sinne eines Vorwärts-Problems konnten Deep Learning Ansätze jedoch auch schon zur Lösung *inverser* nicht-linearer Probleme adaptiert werden (siehe z.B. [17][18][19]). Dies umfasst z.B. die Rekonstruktion physikalischer Parameter in Differenzialgleichungen aus Messdaten oder die Rekonstruktion physikalischer Felder aus komplementären Teildaten. Diese Problemklasse ist für klassische numerische Verfahren (wenn überhaupt) nur mit signifikant höherem Aufwand zugänglich. Unabhängig davon, ob PINNs mittelfristig belastbare Effizienz- und Genauigkeitsvorteile bei der Lösung von Vorwärts-Problemen zeitigen, steht schon jetzt fest, dass sie insbesondere im Bereich der inversen Probleme neue Wege im Computational Engineering zu beschreiten erlauben. Ein weiteres Forschungsfeld wird auch weiterhin die Kombination von Deep Learning und klassischen Ansätzen sein [10].

Literatur

[1] Bellman, R.E. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ (1961)

[2] David Anton, Sensitivitäts- und Genauigkeitsanalysen von Deep-Learning-Ansätzen zur Lösung von partiellen Differentialgleichungen, Masterarbeit, IRMB, TU Braunschweig (2020)

https://doi.org/10.24355/dbbs.084-202109031135-0

[3] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *J. Comput. Phys.*, vol. 378, pp. 686–707, https://doi.org/10.1016/j.jcp.2018.10.045 (2019)
[4] M. Raissi, P. Perdikaris and G. E. Karniadakis: Physics Informed Deep Learning (Part I): Data-driven Discovery of Nonlinear Partial Differential Equations, https://arxiv.org/abs/1711.10561 (2017)
[5] M. Raissi, P. Perdikaris and G. E. Karniadakis: Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations, https://arxiv.org/abs/1711.10566 (2017)
[6] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning : A survey,” *JMLR*, vol. 18, pp. 1–43 (2018)
[7] S. Wang, Y. Teng, and P. Perdikaris, “Understanding and Mitigating Gradient Pathologies in Physics-Informed Neural Networks,” *arXiv Preprint*, p. 28, doi: arXiv:2001.04536v1 [cs.LG] (2020)
[8] G. Cybenko. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems, 2(4):303–314 (1989)
[9] K. Hornik, M. Stinchcombe und H. White. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5):359–366 (1989)
[10] Atsuya Oishi, Genki Yagawa, Computational mechanics enhanced by deep learning, Computer Methods in Applied Mechanics and Engineering, Volume 327, pp. 327-351, https://doi.org/10.1016/j.cma.2017.08.040 (2020)
[11] Juyeon Heo, Sunghwan Joo, Taesup Moon, Fooling Neural Network Interpretations via Adversarial Model Manipulation, https://arxiv.org/abs/1902.02041v3 (2019)
[12] D. O. Hebb, The organization of Behavior – a Neuropsychological Theory, John Wiley & Sons (1949)
[13] Arthur L. Samuel, Some studies in machine learning using the game of checkers, IBM J. Res. Development, Band 3, S. 210–229 (1959), auch in Ed Feigenbaum, Julian Feldman Computers and Thought, McGraw Hill (1961)
[14] https://en.wikipedia.org/wiki/Machine_learning (zuletzt zugängig am 25.04.2021)
[15] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, doi: 10.1038/nature14539 (2015)
[16] D. J. C. MacKay, Information Theory, Inference, and Learning Algorithms. Cambridge University Press, Version 7.2 (2005)
[17] A. Lucas, M. Iliadis, R. Molina and A. K. Katsaggelos, "Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods," in IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 20-36, Jan. 2018, doi: 10.1109/MSP.2017.2760358.
[18] M. Genzel, J. Macdonald and M. März: Solving Inverse Problems With Deep Neural Networks - Robustness Included?, https://arxiv.org/abs/2011.04268 (2020)
[19] H. Li, J. Schwab, S. Antholzer and M. Haltmeier, NETT: solving inverse problems with deep neural networks, https://doi.org/10.1088/1361-6420/ab6d57 (2020)

https://doi.org/10.24355/dbbs.084-202109031135-0